

3717.

Rule  
1.126

A system for retrieving object references in a thread stack comprising:  
means for retrieving an object data structure from the thread stack;  
means for extracting an object reference from one part of the data structure; and  
means for extracting a reference to the next data structure in the thread stack  
from another part of the data structure, the extracted object references forming a root  
set of object references.

#### REMARKS

The present invention concerns the formulation of the root set of object references that are to be marked and collected from the memory of a virtual machine.

Independent claims 1, 11 and 20 (new claims 1, 9, and 17) have been amended to clarify their distinction with respect to 35 USC 102 over USP 6,314,436 to Houldsworth. In particular the terms 'thread stack' and 'root set' have been included to increase clarity of the claims. Furthermore, these amendments to new claims 1, 9, and 17 distinguish these claims (and the remaining claims, which are dependent upon these independent claims) from Houldsworth USP 6,314,436, which does not address the formulation of root set object references in the thread stack, but merely goes on to deal with the garbage collection aspect of the root set.

Houldsworth discloses a marking scheme for tracing objects in a heap that are to be garbage collected. At col 5 line 46 to 50 of Houldsworth, the main procedure of Figure 6 is described, the first procedure being to select the first item in a root set of objects, at 601. Houldsworth goes on to describe the root set as comprising stacks, global objects, and generally any stored data entities that are capable of being referenced from outside the heap. The selected first object and then subsequent objects of the root set are marked in the heap in preparation for collection. Nowhere does Houldsworth explain

how the root set is defined, thereby leaving one to assume that it is probably by means of the prior art conservative scan as described in on page 2 of the present application.

It is noted that the stack mentioned at col 5 line 48 in Houldsworth is the thread stack. The present invention manages objects by means of linking them, so that the root set of objects is always to be found in the thread stack. The advantage of the present invention includes the advantage that linked objects are always dealt with by using the thread stack no matter how the virtual machine is compiled whereas a non-linked object may or may not be dealt with by the thread stack and therefore be difficult to garbage collect. By contrast, the prior art root set of objects comprises non-linked objects which may be found on the thread stack, as global objects or any stored data entity. The conservative scan method which finds the root set only eliminates the data which are definitely not root set objects and is only an approximation of the root set. The root set of objects of the invention comprises linked objects which are only found on the thread stack. This set is much more accurately defined than the root set of the conservative scan of the prior art.

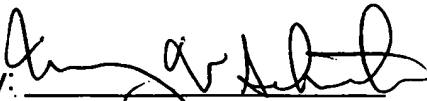
It is believed that the newly amended claims 1-17 overcome the Examiner's objections based on Houldsworth and place the application in condition for allowance, which is respectfully solicited.

Attached hereto is a marked-up version of the changes made to the specification and claims by the current amendment. The attached page is captioned "**Version with markings to show changes made**".

The Examiner is invited to telephone the undersigned attorney at the number appearing below to discuss any issue remaining unresolved to the Examiner's full satisfaction.

Authorization is hereby granted to deduct any applicable USPTO fees arising from this amendment from our deposit account 09-0468.

Respectfully submitted,

By: 

Manny W. Schechter

Registration No. 31,722

IBM Corporation  
Intellectual Property Law Dept.  
P.O. Box 218  
Yorktown Heights, New York 10598  
(914) 945-3252.

**VERSION WITH MARKINGS TO SHOW CHANGES MADE**

**In the claims:**

1. A method of retrieving object references in a thread stack comprising:  
retrieving an object data structure from the thread stack;  
extracting an object reference from one part of the data structure; and  
extracting a reference to the next data structure in the thread stack from another  
part of the data structure, the extracted object references forming a root set of object  
references.
2. A method as claimed in claim 1 further comprising retrieving the next data  
structure and retrieving the corresponding object reference and corresponding next  
data structure reference.
3. A method as claimed in claim 2 further comprising retrieving all linked data  
structures in the thread stack.
4. A method as claimed in claim 3 further comprising retrieving a last data structure  
having no next data structure reference.
5. A method as claimed in claim 4 further comprising retrieving the first object data  
structure in the thread stack referenced by a first object data structure pointer.
- (6. A method as claimed in claim 5 further comprising using the retrieved object  
references to define a root set of objects.)

6 (7). A method as claimed in claim 5 (6) comprising:

defining a reachable set of objects as all objects referenced directly or indirectly by the root set objects.

7 (8). A method as claimed in claim 6(7) further comprising identifying all objects within the process and reclaiming the memory space of all non-reachable objects.

8 (9). A method as claimed in claim 6 or 7 (7 or 8) further comprising moving reachable objects so that they are contiguous in memory and updating all object references in the thread stack by tracing through the chain of object data structures.

9 (11) A method of managing a object in a thread stack based garbage collected virtual machine (process) comprising:

storing an object data structure in the thread stack comprising a reference to the object and a reference to a previously stored object data structure in the stack;

whereby the object data structure and the previously stored object data structure form a root set of data object structures.

10 (12). A method as claimed in claim 9 (11) further comprising linking the object data structure to the previously stored object data structure.

11 (13). A method as claimed in claim 10 (12) further comprising:

storing a variable pointing to the previously stored object data structure at the top of the stack;

using the variable when storing a new object data structure; and  
updating the variable with the new object data structure reference.

12 (14). A method as claimed in claim 11 (13) further comprising:

saving the variable pointer;

storing the object data structure;

updating the variable with the reference to the latest stored object data structure;

performing the process ; and  
restoring the stack pointer.

13 (15). A method as claimed in claim 9 (11) further comprising:  
retrieving an object data structure and extracting the associated object reference  
and data structure reference;  
using the associated data structure reference to retrieve the previously stored  
object data structure;  
retrieving all the object references in the stack by tracing through the chained of  
object data structures.

(16. A method as claimed in claim 15 further comprising using the retrieved object  
pointers to identify a root set of objects.)

14 (17). A method as claimed in claim 13 (16) comprising:  
identifying all objects referenced directly or indirectly by the root set objects and  
marking the root set and all referenced objects as reachable.

15 (18). A method as claimed in claim 14 (17) further comprising identifying all  
objects within the process and reclaiming the memory space of all non-reachable  
objects.

16 (19). A method as claimed in claim 15 (18) further comprising moving  
reachable objects in process memory so that they are contiguous and updating all  
object references in the stack by tracing through the chain of object data structures.

17 (20). A system for retrieving object references in a thread stack comprising:  
means for retrieving an object data structure from the thread stack;  
means for extracting an object reference from one part of the data structure; and  
means for extracting a reference to the next data structure in the thread stack

from another part of the data structure, the extracted object references forming a root set of object references.

